

CLAIMS

What is claimed is:

- 1 1. A method comprising:
2 receiving a first program unit in a parallel computing environment, the first
3 program unit including a memory update operation to be performed
4 atomically, the memory update operation having an operand, the operand
5 being of a data-type and of a data size; and
6 translating the first program unit into a second program unit, the second program
7 unit to associate the memory update operation with a set of one or more
8 low-level instructions upon determining that the data size of the operand is
9 supported by the set of low-level instructions, the set of low-level
10 instructions to ensure atomicity of the memory update operation.
- 1 2. The method of claim 1 wherein to associate the memory update operation with the
2 set of low-level instructions comprises:
3 enclosing the memory update operation in a callback routine; and
4 referencing the callback routine from a routine that references the set of low-level
5 instructions.
- 1 3. The method of claim 1 wherein the set of low-level instructions encapsulate the
2 memory update operation.
- 1 4. The method of claim 1 further comprising translating the first program unit into a
2 third program unit upon determining that a second set of one or more low-level

3 instructions support the memory update operation for the data-type and the data size of
4 the operand, the second set of low-level instructions for performing the memory update
5 operation atomically.

1 5. The method of claim 1 further comprising translating the first program unit into a
2 third program unit upon determining that a second set of low-level instructions does not
3 support the memory update operation for the data-type and the data size of the operand
4 and that the set of low-level instructions does not support the data size of the operand, the
5 third program unit to associate the memory update operation with a set of locking
6 instructions.

1 6. The method of claim 1 wherein the set of low-level instructions for ensuring
2 atomicity is a compare-and-swap instruction.

1 7. A method comprising:
2 receiving a first program unit, the first program unit including a memory update
3 operation to be performed atomically, the memory update operation
4 indicating an operand and an operator, the operand being of a data-type
5 and a data size;
6 translating the first program unit into a second program unit upon determining that
7 a first set of one or more low-level instructions support the memory update
8 operation for the data-type and the data size of the operand, the first set of
9 low-level instructions for performing the memory update operation
10 atomically;

11 translating the first program unit into a third program unit, the third program unit
 12 to associate the memory update operation with a second set of one or more
 13 low-level instructions upon determining that the data size of the operand is
 14 supported by the second set of low-level instructions, the second set of
 15 low-level instructions to ensure atomicity of the memory update operation;
 16 and
 17 translating the first program unit into a fourth program unit upon determining that
 18 the first set of low-level instructions does not support the memory update
 19 operation for the data-type and the data size of the operand and that the
 20 second set of low-level instructions does not support the data size of the
 21 operand, the fourth program unit to associate the memory update operation
 22 with a set of locking instructions.

1 8. The method of claim 7 wherein the second set of low-level instructions
 2 encapsulate the memory update operation.

1 9. The method of claim 7 wherein associating the second set of instructions to the
 2 memory update operation comprises:
 3 enclosing the memory update operation in a callback routine; and
 4 referencing the callback routine from a routine that references the second set of
 5 low-level instructions.

1 10. The method of claim 7 wherein the second set of low-level instructions is a
 2 compare-and-swap instruction.